ĐẠI HỌC HUẾ
TRƯỜNG ĐẠI HỌC KHOA HỌC
KHOA CÔNG NGHỆ THÔNG TIN

TIỂU LUẬN
XỬ LÝ NGÔN NGỮ TỰ NHIÊN

SỬA LỖI CHÍNH TẢ TIẾNG VIỆT SỬ DỤNG MÔ HÌNH BILSTM

Nhóm sinh viên thực hiện:

1. NGUYỄN HOÀI NAM (NT)

2. LÝ NHẬT PHƯƠNG

3. NGUYỄN NGỌC QUANG HUY

Tên học phần: **XỬ LÝ NGÔN NGỮ TỰ NHIÊN - NHÓM 1**

Mã học phần: **2023-2024.1.TIN4633.001**

Giáo viên hướng dẫn: **T.S ĐOÀN THỊ HỒNG PHƯỚC**

Huế, 01 - 2024

Mục lục

Danh sách hình vẽ

LỜI MỞ ĐẦU

Trong thời đại công nghệ ngày nay, sự giao tiếp trực tuyến ngày càng trở nên quan trọng, đặc biệt là khi sử dụng tiếng Việt trong các bài viết, tin nhắn, hay tài liệu. Tuy nhiên, một vấn đề thường xuyên gặp phải là lỗi chính tả, điều này không chỉ gây khó khăn cho người viết mà còn tạo ra hiểu lầm khi đọc. Để giải quyết vấn đề này, chúng ta cần những công cụ hỗ trợ mạnh mẽ.

Mặc dù các nghiên cứu trước đây đã đạt được những kết quả đáng chú ý nhưng vẫn còn tồn tại những thách thức. Và trong bài tiểu luận này, chúng tôi đề xuất phương pháp sử dụng mô hình Bi-directional Long Short-term Memory (BiLSTM) để sửa lỗi chính tả tiếng Việt.

BiLSTM là một mô hình mạng nơ-ron thần kinh sâu có khả năng hiểu được ngữ cảnh từ cả hai hướng, giúp cải thiện khả năng học và dự đoán. Mô hình này có khả năng nắm bắt được ngữ nghĩa của câu đang sửa lỗi chính tả và đề ra từ sửa sai phù hợp nhất.

Chúng tôi đã tìm kiếm một bộ dữ liệu tiếng Việt lớn từ các báo điện tử. Sau đó, chúng tôi tiến hành xây dựng một hệ thống tạo lỗi chính tả tự động để tạo ra tập dữ liệu cho việc huấn luyện mô hình, giúp đánh giá mô hình một cách nhanh hơn. Kết quả thử nghiệm cho thấy phương pháp của chúng tôi đạt được hiệu suất đáng khích lệ với tỉ lệ hơn 90 phần trăm.

Bài tiểu luận này không chỉ nhằm mục tiêu giải quyết vấn đề thực tế mà còn đề xuất một phương pháp hiệu quả sử dụng mạng nơ-ron trong việc xử lý ngôn ngữ tự nhiên Tiếng Việt. Chúng tôi hy vọng rằng nghiên cứu này sẽ đóng góp vào việc nâng cao chất lượng giao tiếp bằng Tiếng Việt trong cộng đồng trực tuyến và các ứng dụng khác.

Chương 1

GIỚI THIỆU

Bài toán sửa lỗi chính tả là một trong những bài toán quan trọng trong lĩnh vực Xử lý ngôn ngữ tự nhiên (NLP). Có nhiều nguyên nhân dẫn đến lỗi chính tả như: lỗi do gõ sai văn bản, lỗi do tạo văn bản bằng công cụ nhận dạng ký tự quang học (OCR), lỗi do sử dụng từ vựng không chính xác...

Tương tự như các ngôn ngữ khác, việc sửa lỗi chính tả tiếng Việt bao gồm hai giai đoạn: giai đoạn phát hiện và giai đoạn sửa lỗi. Giai đoạn đầu tiên chịu trách nhiệm xác định lỗi chính tả trong văn bản. Để khắc phục những lỗi này, giai đoạn 2 phải đưa ra đề xuất ứng viên. Vì mỗi ngôn ngữ đều có những đặc điểm riêng nên chi tiết từng giai đoạn trong mỗi ngôn ngữ cũng khác nhau. Trong tiếng Anh, các từ trong văn bản có thể được phân tách dễ dàng bằng dấu phân cách nhưng trong tiếng Việt (tương tự như tiếng Nhật, tiếng Trung. . .) việc nhận dạng từ rất phức tạp vì mỗi từ tiếng Việt gồm nhiều âm tiết khác nhau [9] [10] và có sự nhầm lẫn giữa các âm tiết. Sự nhầm lẫn này là trở ngại cho hai giai đoạn của hệ thống kiểm tra chính tả tiếng Việt. (**Âm tiết** là đơn vị cơ bản trong ngôn ngữ, được tạo thành từ một hoặc một nhóm các âm thanh ngắn hoặc dài. Mỗi từ trong tiếng Việt được chia thành các âm tiết, và cách các âm tiết này được sắp xếp cùng nhau trong một từ sẽ tạo ra cấu trúc từ đó. Ví dụ, trong từ "hoa quả"có hai âm tiết: "hoa"và "quả". Trong một số trường hợp, âm tiết có thể chỉ có một nguyên âm hoặc kết hợp giữa nguyên âm và phụ âm. Tham khảo tại `https://monkey.edu.vn/ba-me-can-biet/giao-duc/hoc-tieng-viet/am-tiet-la-gi`)

Trong kỉ nguyên công nghệ số hiện nay, sự chính xác và rành mạch của văn bản đóng vai trò quan trọng trong việc truyền tải thông tin. Những lỗi chính tả, lỗi gõ sai

văn bản lúc này sẽ gây ra rất nhiều trở ngại lớn, khi thông tin được truyền đi nhanh chóng chỉ với một cú nhấn chuột, gây hiểm nhầm cho người đọc hoặc hoặc tệ hơn là lan truyền thông tin sai lệch.

Trong hầu hết các trường hợp, lỗi trong tiếng Việt có hai loại: lỗi gõ nhầm (mistyped errors) và lỗi chính tả (misspelled errors) [12] [8]. Lỗi gõ nhầm là lỗi xảy ra trong quá trình gõ chữ. Phần lớn những sai lầm này là do những hành động vô ý của người đánh máy như bấm nhầm phím giữa hai ký tự liền kề trên bàn phím. Hơn nữa, những lỗi này thường dừng lại ở cấp độ âm tiết và chúng có thể được phát hiện nếu người đánh máy xem xét kỹ văn bản. Lỗi gõ nhầm có thể được phân thành hai loại nhỏ hơn: lỗi không phải từ (non-word errors) và lỗi từ thực (real-word errors). Lỗi không phải từ có nghĩa là từ đó hoàn toàn không tồn tại trong từ điển. Mặt khác, lỗi từ thực là lỗi mà những từ vẫn còn trong từ điển nhưng được sử dụng sai ngữ cảnh.

Lỗi chính tả là lỗi mà người đánh máy không nhận ra là sai. Loại lỗi này xảy ra do lỗi phát âm vùng miền hoặc do độ khó của một số từ tiếng Việt. So với lỗi gõ sai, lỗi gõ sai chính tả khó phát hiện hơn vì chúng ta không chỉ cần dựa vào ngữ cảnh mà còn phải có kiến thức về ngôn ngữ chuẩn để phát hiện các lỗi này. Bảng 1.1 trình bày một số ví dụ về lỗi chính tả trong tiếng Việt.

Bảng 1.1: Một số ví dụ về lỗi chính tả Tiếng Việt [12].

Lỗi Non-word	Original: Hôm nay tôi đi **hocj**. Correct: Hôm nay tôi đi **học**.
Lỗi Real-word	Original: **Sướng** còn đọng trên lá. Correct: **Sương** còn đọng trên lá.
Lỗi chính tả	Original: Tôi **muống** ăn thịt. Correct: Tôi **muốn** ăn thịt.
Lỗi phát âm	Original: Tôi không thích cá **cượt**. Correct: Tôi không thích cá **cược**.
Lỗi dấu	Original: Ngôi nhà đầy **ấp** tiếng cười. Correct: Ngôi nhà đầy **ắp** tiếng cười.
Lỗi teencode	Original: Em **ko** muốn có **ck**. Correct: Em **không** muốn có **chồng**.

Sửa lỗi chính tả là vấn đề nhận được rất nhiều sự quan tâm của cộng đồng xử lý ngôn ngữ tự nhiên. Hãy cùng xem qua một số cách tiếp cận bài toán này.

Chương 2

CÁC CÁCH TIẾP CẬN

Cách tiếp cận dựa trên từ điển là phương pháp đơn giản nhất để kiểm tra chính tả. Đầu tiên, những âm tiết không xuất hiện trong từ điển được xác định là có lỗi. Cách thứ hai, tạo ra bộ các âm tiết ứng cử viên dựa trên sự giống nhau với âm tiết hiện tại, nếu có bất kỳ ứng cử viên nào trong bộ ứng cử viên, kết hợp với âm tiết xung quanh có thể tạo ra một từ trong từ điển thì âm tiết hiện tại được xác định là lỗi. Rõ ràng, phương pháp này có hạn chế trong việc phát hiện lỗi theo âm tiết đơn và lỗi âm tiết tồn tại trong từ vựng.

Đã có rất nhiều nghiên cứu tiếp cận vấn đề này bằng cách áp dụng các mô hình ngôn ngữ thống kê ([3], [4], [7], [11], [13]). Các cách tiếp cận trước đây có thể được chia chủ yếu thành hai loại. Một bên sử dụng các mô hình ngôn ngữ thống kê truyền thống và bên kia sử dụng máy học.

- Nguyễn Đức Hải và Nguyễn Phạm Hạnh Nhi [3] áp dụng phân tích câu vào mô hình kiểm tra chính tả tiếng Việt. Câu đầu vào được phân tích bằng thuật toán Earley, nếu câu có lỗi chính tả thì sẽ không được phân tích cú pháp chính xác. Logic đằng sau mô hình này là các âm tiết tạo nên câu không thể phân tích được có thể được xác định là lỗi chính tả. Tuy nhiên, mô hình này có ba hạn chế. Thứ nhất, độ phức tạp của thuật toán Earley rất tốn kém. Với các âm tiết trong câu đầu vào, độ phức tạp của thuật toán là $O(n^3)$ dẫn đến khó khăn trong việc xử lý các tài liệu dài. Thứ hai, tiếng Việt có khoảng 3000 quy tắc, việc thu thập và sử dụng hết các quy tắc là một thách thức rất lớn. Cuối cùng, có sự mơ hồ trong tiếng Việt. Kết quả là có thể có một câu viết sai chính tả nhưng vẫn có thể phân tích được.

- Năm 2008, Nguyễn Hứa Phùng và cộng sự [4], đã đề xuất một phương pháp thống kê sử dụng POS Bigram (Part Of Speech Bigram) để phát hiện các âm tiết bị nghi ngờ. Các thuật toán Minimum Edit Distance và SoundEx đã được áp dụng để tạo ra các đề xuất đề xuất trong giai đoạn sửa lỗi.
- Năm 2015, Nguyễn Thị Xuân Hương và cộng sự [7], đã phát triển mô hình ngôn ngữ large N-gram để sửa lỗi chính tả tiếng Việt. Mô hình này được tạo ra dựa trên phương pháp thống kê. Một tập dữ liệu lớn không có nhãn được sử dụng để tìm hiểu ngữ cảnh của âm tiết. Cụ thể, điểm N-gram cho mỗi âm tiết trong tập ứng viên được tính dựa trên tần suất xuất hiện trong unigram, bigram và trigram. Mô hình tạo tập ứng viên dựa trên việc thay đổi các ký tự trong âm tiết tương ứng với lỗi đánh máy, lỗi phụ âm, v.v. Âm tiết hiện tại được coi là lỗi nếu một âm tiết trong tập ứng cử viên có điểm N-gram cao hơn âm tiết hiện tại. Cách tiếp cận này hiện là công nghệ tiên tiến nhất với khoảng 94 phần trăm điểm F1 trên dữ liệu thử nghiệm của họ.
- Bằng cách phát hiện và sửa lỗi chính tả, Nguyễn Hồng Vũ và công sự [11], đã phát triển phương pháp chuẩn hóa tweet tiếng Việt dựa trên mô hình ngôn ngữ với từ điển và cấu trúc từ vựng tiếng Việt. Các tweet có lỗi chính tả sẽ được phát hiện dựa trên sự giống nhau của các từ. Sau khi phát hiện lỗi, hệ thống sửa lỗi bằng cách liên kết các nguyên âm, phụ âm và đánh giá bằng mô hình ngôn ngữ. Trong mô hình này, tác giả cải tiến dựa trên mô hình Dice và SRILM (mô hình ngôn ngữ).
- Sửa lỗi chính tả là một bước quan trọng trong việc cải thiện độ chính xác của văn bản do OCR tạo ra. (OCR là viết tắt của "Optical Character Recognition,"có nghĩa là "Nhận diện ký tự quang học". Đây là một công nghệ sử dụng để chuyển đổi hình ảnh chứa văn bản in hoặc viết tay thành văn bản có thể xử lý được trên máy tính.) Đối với lỗi OCR của tiếng Việt, Nguyễn Quốc Dũng và công sự [13], đã phát triển một phương pháp tạo và chấm điểm các ứng viên sửa lỗi dựa trên các đặc điểm ngôn ngữ. Các lỗi chính tả sẽ được phát hiện dựa trên các từ điển unigram, bigram, trigram. Sau giai đoạn phát hiện, một tập ứng cử viên cho mỗi lỗi âm tiết sẽ được tạo ra bằng cách áp dụng các toán tử chèn, xóa và thay thế. Những thí sinh có điểm cao được tính dựa trên các đặc điểm ngôn ngữ như Độ tương tự âm tiết, tần số Bigram, ...

- Năm 2018, Nguyễn Hà Thanh và công sự [8], đề xuất phương pháp học sâu để giải quyết lỗi chính tả phụ âm tiếng Việt. Để xác định và sửa các vị trí lỗi, mô hình sử dụng mã hóa hướng sai chính tả và kiến trúc LSTM xếp chồng hai chiều. Đây là nguồn cảm hứng chính cho bài tiểu luận này.
- Ngoài ra, sửa lỗi chính tả có thể được coi là một vấn đề dịch một chuỗi lỗi chính tả sang một chuỗi đã sửa. Loại vấn đề này có thể được giải quyết bằng các phương pháp điển hình được sử dụng cho dịch máy. Một số nhà nghiên cứu đã áp dụng mô hình Neural Machine Translation (NMT) ([6], [2], [1]) để sửa lỗi chính tả trong các ngôn ngữ phổ biến như tiếng Anh và tiếng Trung. Kết quả khả quan của họ chứng tỏ đây là giải pháp khả thi cho vấn đề sửa lỗi chính tả. Tuy nhiên, một trong những thách thức với NMT là vấn đề hết từ vựng. Tăng kích thước từ vựng của mô hình là một cách đơn giản để giải quyết vấn đề này, tuy nhiên, nếu từ vựng quá lớn thì kích thước của việc nhúng vectơ sẽ quá cao. Nó làm tăng thời gian tính toán và tăng thêm độ phức tạp cho việc huấn luyện mô hình.

Những mô hình truyền thống này tìm hiểu bối cảnh bằng cách đào tạo trên một tập dữ liệu lớn. Tuy nhiên, những phương pháp này có hạn chế: ngữ cảnh của một âm tiết chỉ có thể được nắm bắt bởi các âm tiết liền kề. (Ví dụ trong bi-gram, một từ sẽ liên kết với 1 từ ở trước và 1 từ ở sau nó). Đối với những câu có từ hai lỗi chính tả trở lên cạnh nhau, mô hình sẽ khó xác định lỗi hơn.

Trong những năm gần đây, việc ứng dụng mô hình học sâu (deep learning) vào kiểm tra chính tả tiếng Việt đang là xu hướng mới được các nhà nghiên cứu quan tâm [8]. Ưu điểm của cách tiếp cận này là ngữ cảnh của âm tiết không bị ràng buộc bởi các âm tiết xung quanh, cho phép mô hình phát hiện lỗi chính tả chính xác hơn. Sở dĩ như vậy là nhờ vào những lí do sau:

- Đặc tính Bidirectional: Mô hình BILSTM có khả năng xem xét cả phía trước và phía sau của mỗi từ trong một câu. Điều này cho phép nó hiểu rõ ngữ cảnh toàn cục của câu, không chỉ giới hạn ở các âm tiết liền kề. Sự hai chiều (bidirectional) này giúp mô hình hiểu rõ hơn mối quan hệ giữa các từ trong câu và có thể cải thiện khả năng sửa lỗi chính tả.
- Khả năng nhận biết ngữ cảnh quan trọng: BILSTM có khả năng học được ngữ cảnh quan trọng trong câu thông qua cơ chế LSTM, nơi thông tin được truyền

từng cấp độ thời gian, giúp nó nhớ và sử dụng thông tin ngữ cảnh một cách hiệu quả. Điều này giúp mô hình nhận biết và sửa lỗi chính tả dựa trên ngữ cảnh toàn cụ hơn.

- Tính linh hoạt của LSTM: LSTM có khả năng xử lý chuỗi dữ liệu và duy trì trạng thái ẩn, giúp nó giữ thông tin về ngữ cảnh của câu qua nhiều từ. Điều này làm cho nó trở nên linh hoạt trong việc xử lý ngữ cảnh của các câu có cấu trúc phức tạp và không bị ràng buộc bởi ngữ cảnh cục bộ.

Đó là lí do tại sao chúng tôi chọn mô hình BiLSTM cho bài tiểu luận này.

Chương 3

PHƯƠNG PHÁP THỰC HIỆN

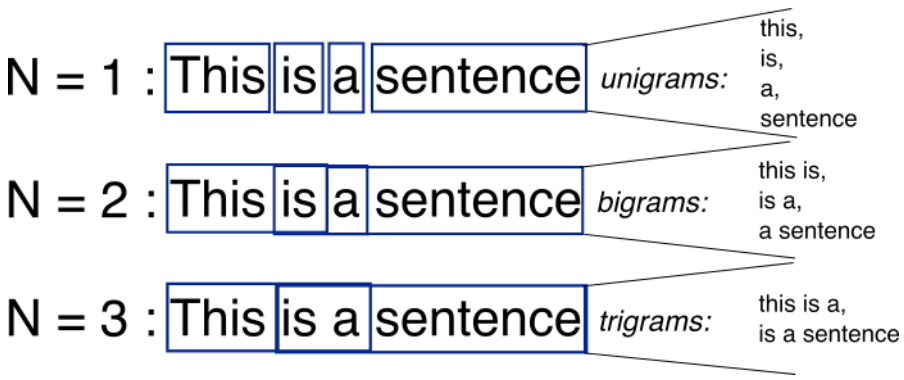
Mặc dù đã đạt được một số kết quả tích cực, hầu hết tất cả các nghiên cứu trước chủ yếu tập trung vào việc sửa một số loại lỗi chính tả nhất định, gây khó khăn khi áp dụng vào thực tế.

Ở bài tiểu luận này, chúng tôi đề xuất một mô hình Bidirectional Long Short-term Memory (BiLSTM) để sửa lỗi chính tả tiếng Việt và đánh giá nó bằng một bộ dữ liệu thực tế. (Tham khảo [8]).

Chúng ta hãy cùng tìm hiểu một số mô hình được sử dụng trong bài tiểu luận.

3.1 N-gram

Mô hình N-gram từ là một mô hình xác suất truyền thống trong ngôn ngữ học. Khung cửa sổ từ được tạo ra có độ dài là n kí tự dùng để tính xác suất xuất hiện của các từ với nhau. Mô hình dự đoán một từ tiếp theo trong chuỗi dựa trên cửa sổ từ trước đó (Hình 3.1).



Hình 3.1: Hình ảnh minh hoạ mô hình N-gram với nhiều mức N khác nhau.
(Nguồn: https://deepai.org)

Công thức chung của N-gram có dạng như sau:

$$P(w_n|w_{n-1},w_{n-2},\ldots,w_1) = \frac{P(w_{n-1},w_{n-2},\ldots,w_1,w_n)}{P(w_{n-1},w_{n-2},\ldots,w_1)} \tag{3.1}$$

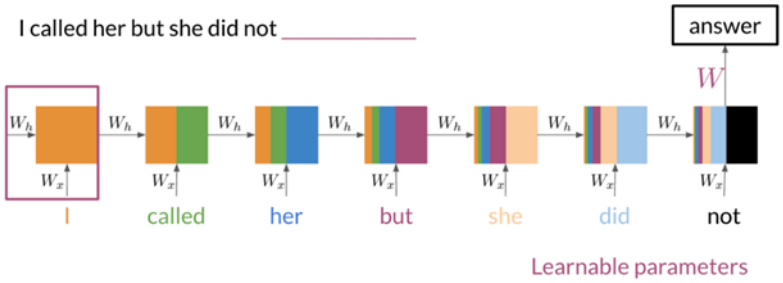
Trong đó :

- $P(w_{n-1},w_{n-2},\ldots,w_1,w_n)$ là xác suất kết hợp của chuỗi từ $w_{n-1},w_{n-2},\ldots,w_1,w_n$ xuất hiện theo thứ tự cho trước.
- $P(w_n|w_{n-1},w_{n-2},\ldots,w_1)$ là xác suất có điều kiện để từ w_n xuất hiện với chuỗi từ cho trước.

Mô hình N-gram là một mô hình đơn giản nhưng hiệu quả cho việc dự đoán từ bằng việc biến đổi các chuỗi từ thành các đại diện xác suất. Tuy nhiên, mô hình này có nhược điểm là chỉ có thể xem xét những chuỗi từ có độ dài nhất định, không thể nắm bắt tốt được ngữ nghĩa của từ hay mối liên hệ giữa các từ trong chuỗi từ dài.

Khi thực hiện làm mịn (smoothing) N-gram để phát hiện từ không có trong tập huấn luyện thì ta có thể làm giảm xác suất của những từ có thể xuất hiện và tăng xác suất của những từ không có ý nghĩa, không xuất hiện trong tập huấn luyện.

3.2 Recurrent Neural Network

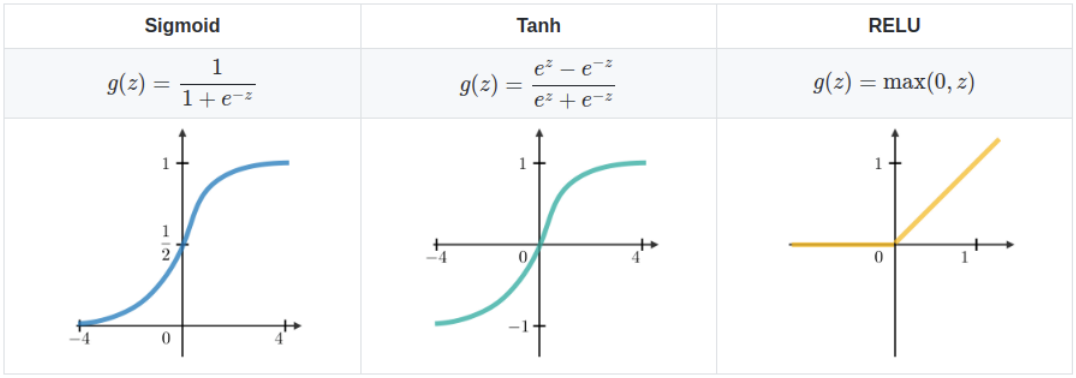


Hình 3.2: Hình ảnh minh hoạ cách thức hoạt động của mạng RNN.
(Nguồn: https://phamdinhkhanh.github.io)

Recurrent Neural Network (RNN) được thiết kế để xử lí thông tin dạng chuỗi, với thứ tự thành phần của thông tin đầu vào được nhấn mạnh. Trong việc xử lí ngôn ngữ tự nhiên, RNN có thể nắm bắt ngữ nghĩa, ngữ cảnh và mối quan hệ giữa cá từ một cách hiệu quả nhờ việc có những liên kết được hình thành và lưu giữ thông tin từ đầu vào ban đầu đến các thành phần thông tin sau này (Hình 3.2).

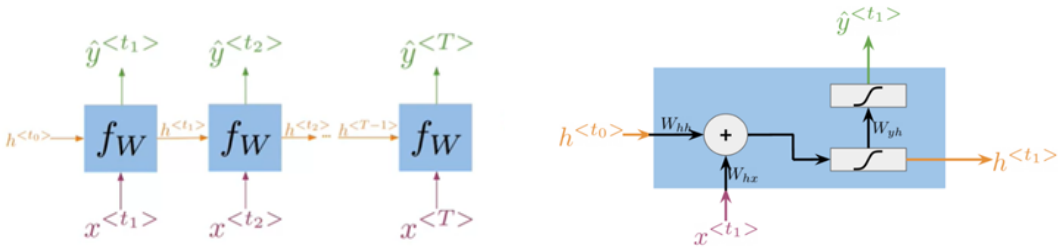
Trong RNN, có một tham số học sẽ lưu lại thông tin về các tính toán về chuỗi thông tin được gọi là trạng thái ẩn (hidden state).

Hidden state và những trọng số đầu ra ở các bước của RNN được quyết định bởi lớp kích hoạt (Hình 3.3) để đưa dữ liệu vào chuỗi sau.



Hình 3.3: Hình ảnh minh hoạ lớp kích hoạt trong RNN.
(Nguồn: https://phamdinhkhanh.github.io)

Kiến trúc mạng RNN cho phép đầu ra từ những chuỗi thông tin trước trở thành đầu vào của những chuỗi thông tin sau, và có cấu trúc như hình 3.4:

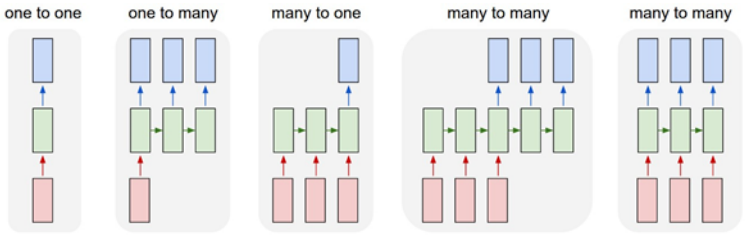


Hình 3.4: Kiến trúc của mạng RNN (Bên trái là kiến trúc tổng quan; Bên phải là cấu tạo của một lớp Feedforward). (Nguồn: https://phamdinhkhanh.github.io)

Trong đó:

- $h^{<t>} = g(W_h[h^{<t-1>}, x^{<t>}] + b_h$: Hidden state của mô hình RNN tại mỗi bước
- $y^{<t>} = g(W_{hh}h^{<t-1>} + W_{hx}x^{<t>} + b_h)$: Đầu ra của mô hình RNN với $x^{<t>}$ là đầu vào

Bài toán sử dụng RNN sẽ được phân loại dựa trên cách $y^{<t>}$ được đưa ra khỏi mô hình từ đầu vào là $x^{<t>}$, bao gồm các loại bài toán như hình 3.5.



Hình 3.5: Mô hình RNN được sử dụng theo các loại bài toán.
(Nguồn: https://phamdinhkhanh.github.io)

Ưu điểm của RNN:

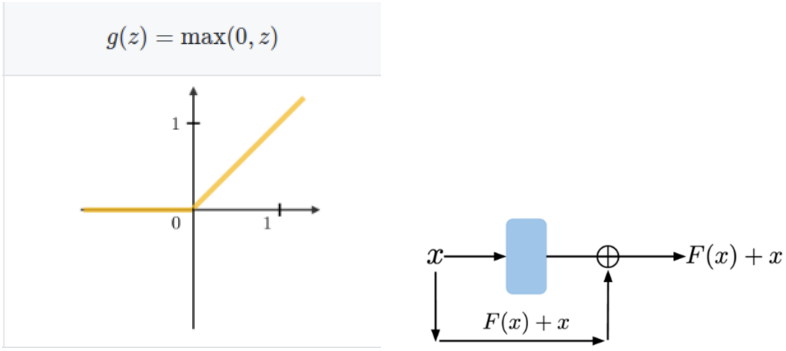
- Giúp nắm bắt được các phụ thuộc từ về mặt ngữ pháp và ngữ nghĩa trong một khoảng ngắn.
- Tốn ít tài nguyên bộ nhớ hơn mô hình N-gram.

Nhược điểm của RNN:

- Gặp khó khăn với các chuỗi thông tin có độ dài lớn.
- Đạo hàm bị triệt tiêu (vanishing gradient), exploding gradient(đạo hàm bùng nổ) do các trọng số trở nên nhỏ đi khi đi qua nhiều lớp, đến khi không còn giá trị sử dụng được hoặc tăng quá mức kiểm soát khi có quá nhiều dữ liệu.

Giải pháp cho vấn đề Vanishing Gradient (Hình 3.6):

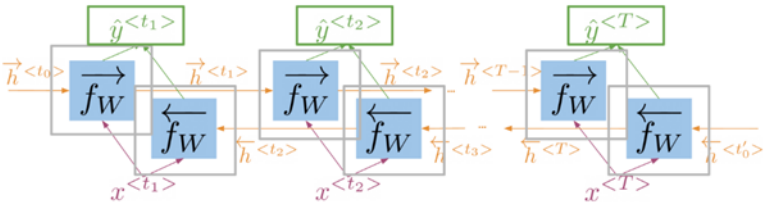
- RNN với hàm kích hoạt ReLu.
- Gradient clipping.
- Skip connection.



Hình 3.6: Giải pháp cho vấn đề Vanishing Gradient (Bên trái là hàm kích hoạt ReLU; Bên phải là skip connection. (Nguồn: https://phamdinhkhanh.github.io)

3.3 Bi-dierctional RNN

Bi-dierctional RNN là mạng RNN nhưng thay vì chỉ di chuyển một chiều dữ liệu từ trái sang phải, thì lúc này Bi-dierctional RNN có hai luồng dữ liệu đi theo hai hướng ngược chiều nhau (Hình 3.7).



Hình 3.7: Hình ảnh mô hình Bi-directional RNN.
(Nguồn: https://phamdinhkhanh.github.io)

Bi-dierctional RNN rất quan trọng, vì nó cho ta biết ngữ cảnh tiếp theo sau từ cần phải dự đoán trong mô hình xử lí ngôn ngữ tự nhiên.

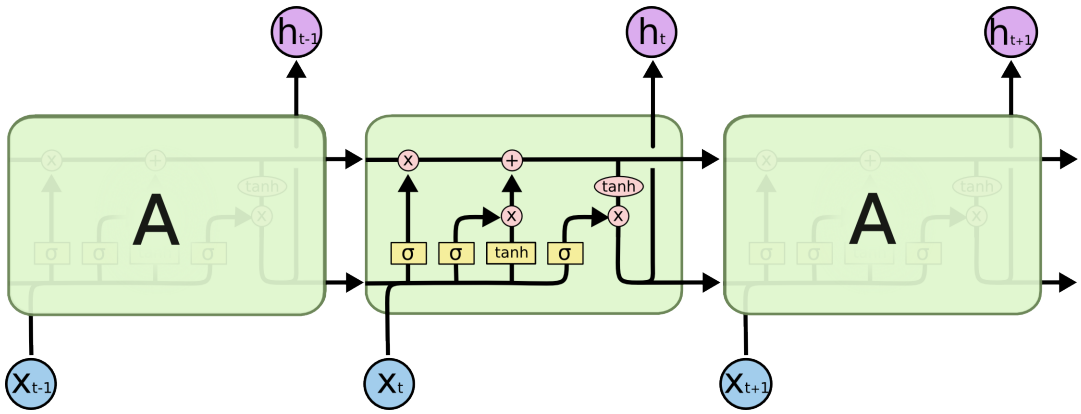
Để có thể dự đoán được $y^{<t>}$, ta cần hidden state của cả hai luồng và kết hợp chúng lại để tạo ra một hidden state mới và tiền hành như RNN thông thường.

3.4 Long Short-term Memory

3.4.1 Giới thiệu mô hình

Mạng Long Short-term Memory (LSTM) là một biến thể của RNN có khả năng học hỏi những phụ thuộc theo thứ tự trong vấn đề dự đoán chuỗi dữ liệu [5].(`https://phamdinhkhanh.github.io/2019/04/22/Ly_thuyet_ve_mang_LSTM.html`)

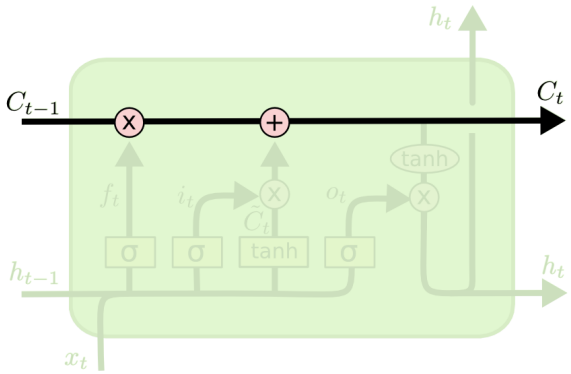
LSTM giải quyết hai vấn đề của RNN là vanishing gradient và exploding gradient bằng cách chọn lựa ra những thông tin cần thiết ở mỗi bước của mô hình và quên những thông tin còn lại. Lúc mô hình có thể lưu lại thông tin ở xa thông tin ở đầu chuỗi (Hình 3.8).



Hình 3.8: Hình ảnh mô hình Long Short-term Memory
(Nguồn: https://colah.github.io)

LSTM gồm một cell state và một hidden state với ba cổng:

- Cổng Input i: cho biết bao nhiêu thông tin được input vào tại một thời điểm nhất định.
- Cổng Forget f: cho biết bao nhiều thông tin cần quên tại một thời điểm.
- Cổng Output o: cho biết bao nhiều thông tin truyền qua tại một thời điểm.
- Cell state: Giúp mô hình giữ lại thông tin quan trọng và quên đi thông tin không quan trọng (Hình 3.9).



Hình 3.9: Hình ảnh Cell state
(Nguồn: https://colah.github.io)

LSTM có khả năng thêm hoặc bớt đi thông tin ở Cell state thông qua các cổng Input, Output và Forget. Cổng (Gate) là một cách để lựa chọn thông tin đi qua mạng LSTM. Chúng được cấu thành từ lớp output sigmoid và một phép toán nhân. Lớp output sigmoid cho ra một số thuộc về giá trị 0 hoặc 1, quyết mỗi thành phần được duyệt qua mạng neural hay không.

Những tham số trong LSTM được tính như sau (Tham khảo [8]):

- Cổng Input:

$$i_t = \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{t-1} + b_{hi}) \tag{3.2}$$

- Cổng Forget:

$$f_t = \sigma(W_{if}x_t + b_{if} + W_{hf}h_{t-1} + b_{hf}) \tag{3.3}$$

- Cổng Cell:

$$g_t = \tanh(W_{ig}x_t + b_{ig} + W_{hg}h_{t-1} + b_{hg}) \tag{3.4}$$

- Cổng Output:

$$o_t = \sigma(W_{io}x_t + b_{io} + W_{ho}h_{t-1} + b_{ho}) \tag{3.5}$$

- Cell State:

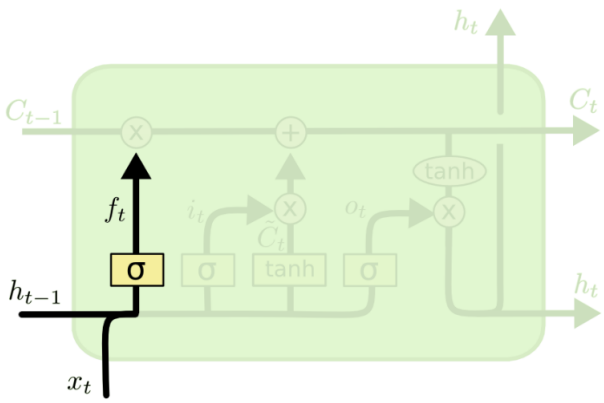
$$c_t = f_t \odot c_{t-1} + i_t \odot g_t \tag{3.6}$$

- Hidden State:

$$h_t = o_t \odot \tanh(c_t) \tag{3.7}$$

3.4.2 Cách thức hoạt động

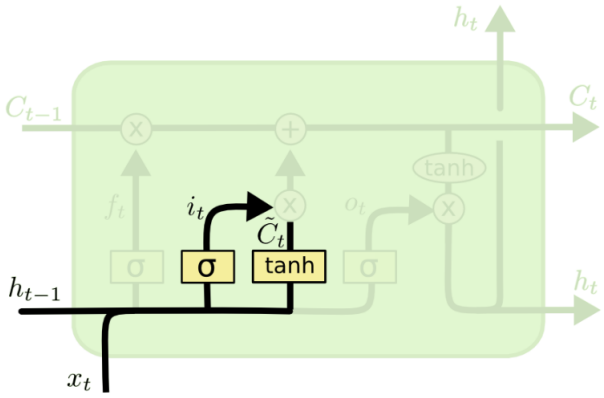
Trong mô hình LSTM, bước đầu tiên phải làm là xác định thông tin nào trong Cell state cần được quên đi (đi qua cổng Forget 3.10) bằng công thức 3.2.



Hình 3.10: Hình ảnh Cổng Forget
(Nguồn: https://colah.github.io)

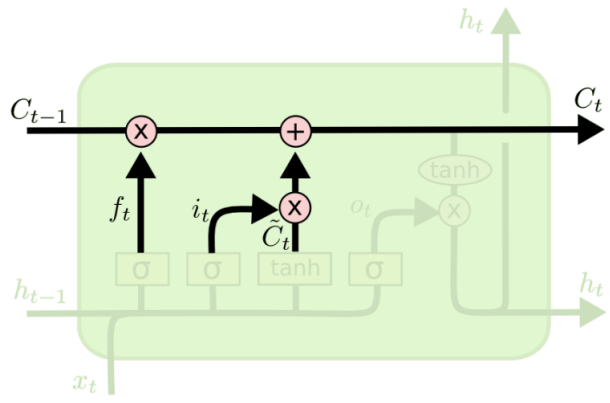
Bước tiếp theo là quyết định thông tin mới được đưa vào Cell state và gồm hai phần như sau (Hình 3.11):

- Một lớp sigmoid gọi là cổng Input sẽ quyết định giá trị nào sẽ được cập nhập theo công thức 3.1.
- Lớp tanh sẽ tạo ra một vector các giá trị ứng viên đề đưa vào Cell state là g_t theo công thức 3.3.



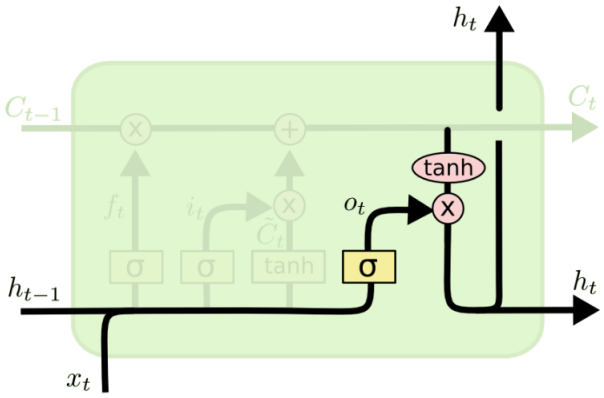
Hình 3.11: Hình ảnh cổng Tạo ra các giá trị ứng viên vào Cell state
(Nguồn: https://colah.github.io)

Cập nhập Cell state (Hình 3.12) cũ là C_{t-1} và Cell state mới C_t bằng cách nhân giá trị cũ với f_t và cộng với giá trị mới nhân cho cổng Input theo công thức 3.5.



Hình 3.12: Hình ảnh cập nhật Cell state
(Nguồn: https://colah.github.io)

Bước cuối cùng là đầu ra của mô hình thông qua Cell state (Hình 3.13), lúc này sẽ đi qua lớp sigmoid để quyết định phần nào trong Cell state được đưa ra theo công thức 3.5. Sau đó Cell state sẽ đi qua lớp tanh để chuyển giá trị về khoảng (-1, 1) và nhân nó với đầu ra của cổng sigmoid theo công thức 3.4.

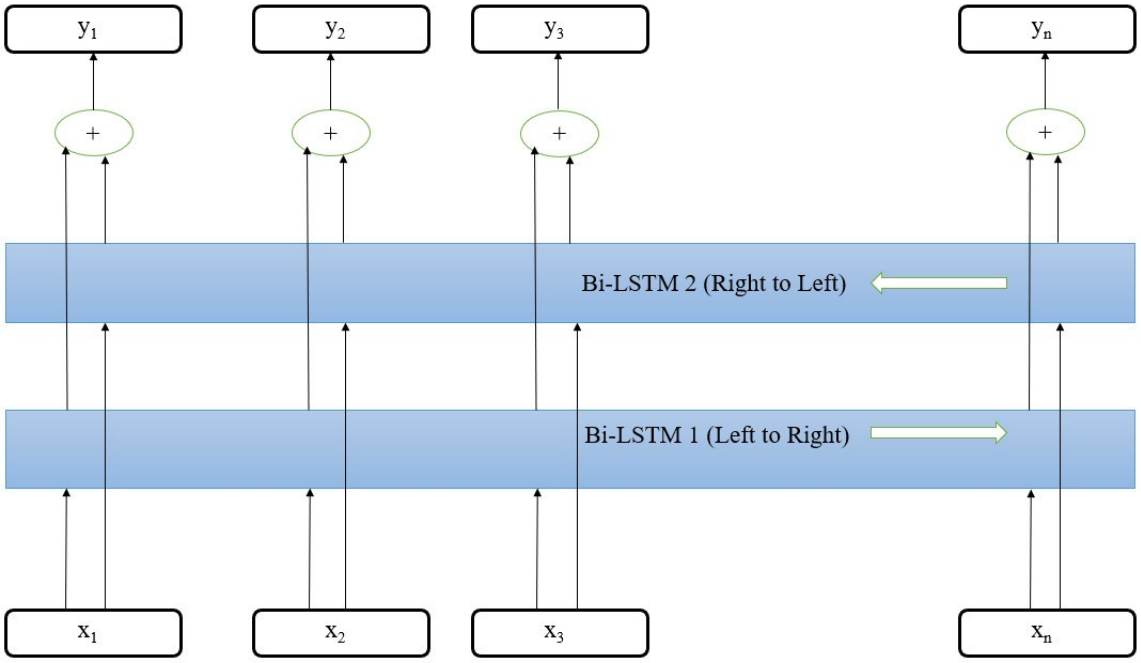


Hình 3.13: Hình ảnh đầu ra của Cell state
(Nguồn: https://colah.github.io)

3.5 Bidirectional-LSTM

Ta có thể sử dụng mạng nơ ron hồi quy theo hai chiều ngược nhau để xử lý một đơn vị RNN sẽ làm như thường lệ, tức là ta sẽ dùng nó để học các tín hiệu đầu vào từ thời điểm ban đầu tới thời điểm kết thúc (đi xuôi). Bên cạnh đó, đơn vị RNN còn lại, ta sẽ đọc theo thứ tự thời điểm từ kết thúc trở lại ban đầu (đi ngược). Sau khi có cả hai kết quả, chúng sẽ được gom lại thành một để có thể dự đoán. Với ý tưởng như vậy, tại một thời điểm bất kỳ, mạng sẽ có được các thông tin trước và sau thời điểm *t* hiện tại.

Do bản chất LSTM là cải tiến của RNN, cho nên ta có thể áp dụng nó và biến nó thành mạng nơ ron dài ngắn song song (BiLSTM). Mỗi LSTM sẽ vẫn có khả năng quên thông tin cũ (cổng quên), lọc thông tin mới (cổng đầu vào), hoặc giấu bớt kết quả (cổng đầu ra) như bình thường. Chính vì vậy, các thông tin từ quá khứ tới tương lai của mạng BiLSTM đều có thể tự học để tự điều chỉnh. Dẫn tới việc với các bài toán mà ta cần biết nhiều hơn về ngữ cảnh hiện tại của nó, thì mạng BiLSTM cho kết quả tốt hơn. Hình 3.14 dưới đây mô tả mạng BiLSTM:



Hình 3.14: Hình ảnh mô tả mạng BiLSTM
(Nguồn: https://colah.github.io)

Chương 4

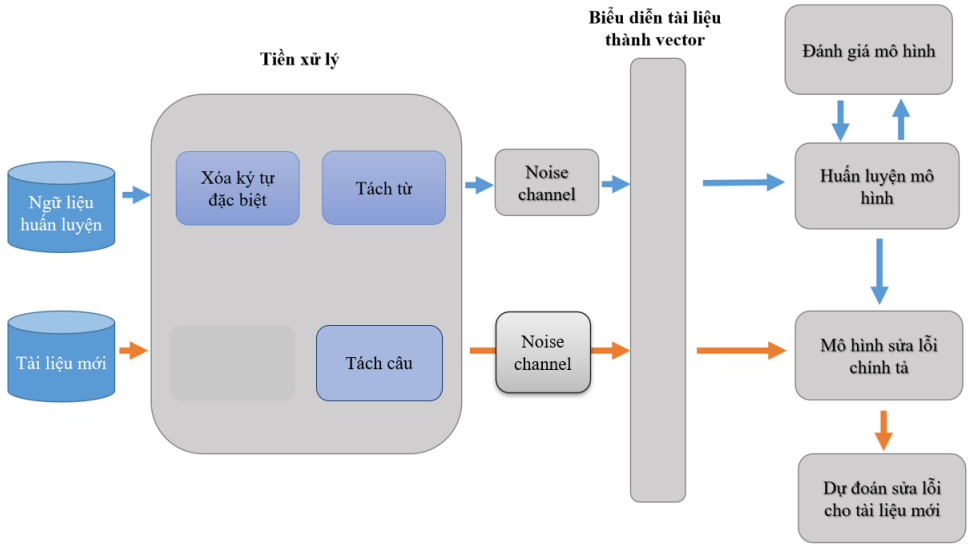
THỬ NGHIỆM VÀ ĐÁNH GIÁ

Trong phần này, chúng tôi thử nghiệm mô hình BiLSTM trên tập dữ liệu VNTC.

4.1 Tổng quan các bước xây dựng

Quá trình thử nghiệm mô hình được thực hiện trên Google Colab. Chúng tôi thực hiện huấn luyện, so sánh và đánh giá dựa trên kết quả của phương pháp với các mô hình trước đây.

Về cơ bản, quy trình giải quyết bài toán sửa lỗi chính tả tiếng Việt của nhóm được thể hiện như hình dưới đây (Hình 4.1):



Hình 4.1: Hình ảnh Mô tả quy trình giải quyết bài toán sửa lỗi chính tả tiếng Việt.

Mô hình xây dựng trong bài toán sửa lỗi chính tả tiếng Việt được đưa ra cần phải đáp ứng những tiêu chuẩn sau:

- Nắm bắt được đúng ngữ pháp, ngữ nghĩa của câu đang xét và quan hệ giữa các từ có trong nó.
- Từ mô hình ngôn ngữ đã được huấn luyện, phát hiện ra được từ có chứa lỗi về mặt ngữ pháp hoặc ngữ nghĩa tiếng Việt.
- Đề xuất ra được từ thay thế từ sai lỗi chính tả đã phát hiện và thay thế nó bằng từ được cho là đúng nhất.
- Mô hình được xây dựng phải có độ chính xác và tính tin cậy cao.

4.2 Dữ liệu đánh giá

4.2.1 Thu thập dữ liệu

Dữ liệu huấn luyện được lấy từ tập dữ liệu VNTC được tổng hợp từ nhiều trang báo điện tử khác nhau như vnexpress.net, tuoitre.vn, thanhnien.vn, nld.com.vn. Các bài báo từ rất nhiều lĩnh vực trong cuộc sống như chính trị xã hội, đời sống, khoa học, kinh doanh, pháp luật, sức khoẻ, thế giới, thể thao, văn hoá, ... được chia thành các tập văn bản nhỏ. Tập dữ liệu được chia thành hai phần là tập huấn luyện và tập kiểm thử.

Link dữ liệu: `https://github.com/duyvuleo/VNTC/tree/master/Data/10Topics/Ver1.1`

Topic	**Topic ID**	**Files**
Chính trị xã hội	XH	5219
Đời sống	DS	3159
Khoa học	KH	1820
Kinh doanh	KD	2552
Pháp luật	PL	3868
Sức khoẻ	SK	3384
Thế giới	TG	2898
Thể thao	TT	5298
Văn hoá	VH	3080
Công nghệ thông tin	IT	2481
Tổng cộng		**33759**

Bảng 4.1: Tệp huấn luyện

Topic	**Topic ID**	**Files**
Chính trị xã hội	XH	7567
Đời sống	DS	2036
Khoa học	KH	2096
Kinh doanh	KD	5276
Pháp luật	PL	3788
Sức khoẻ	SK	5417
Thế giới	TG	6716
Thể thao	TT	6667
Văn hoá	VH	6250
Công nghệ thông tin	IT	4560
Tổng cộng		**50373**

Bảng 4.2: Tệp kiểm thử

Bảng 4.3: Tổng số tệp văn bản có trong tệp dữ liệu

4.2.2 Tiền xử lí văn bản

Do nếu sử dụng tất cả văn bản trong tập huấn luyện sẽ làm cho việc huấn luyện mô hình trở nên kéo dài, nhóm đã quyết định cắt giảm các chủ đề trong tập huấn luyện và kiểm thử nhằm giảm số lượng dữ liệu phải xử lí (Cắt giảm chủ đề Đời sống, Thế Giới, Văn Hoá, Công Nghệ Thông Tin).

Topic	Topic ID	Files
Chính trị xã hội	XH	1499
Khoa học	KH	1499
Kinh doanh	KD	1499
Pháp luật	PL	1499
Sức khoẻ	SK	1499
Thể thao	TT	1499
Tổng cộng		**8994**

Bảng 4.4: Tệp huấn luyện

Topic	Topic ID	Files
Chính trị xã hội	XH	1499
Khoa học	KH	1499
Kinh doanh	KD	1499
Pháp luật	PL	1499
Sức khoẻ	SK	1499
Thể thao	TT	1499
Tổng cộng		**8994**

Bảng 4.5: Tệp kiểm thử

Bảng 4.6: Tổng số tệp văn bản sau khi cắt giảm

Dữ liệu huấn luyện sau khi cắt giảm sẽ được chia thành từng câu một, loại bỏ đi các kí tự đặc biệt và kí tự phi Latin (không thuộc alphabet ở dưới) bằng cách quy định chỉ các kí tự trong danh sách mới được phép có trong văn bản, nếu không thì sẽ bị xóa (Hình 4.2).

```
alphabet = ^[ _abcdefghijklmnopqrstuvwxyz0123456789áàảãạâấầẩẫậăắằẳẵặ
óòỏõọôốồổỗộơớờởỡợéèẻẽẹêếềểễệúùủũụưứừửữựíìỉĩịýỳỷỹỵđ!\"\',\-\.:;?_\(\)]+$
```

```
import re
alphabet = '^[ _abcdefghijklmnopqrstuvwxyz0123456789áàảãạâấầẩẫậăắằẳẵặóòỏõọôốồổỗộơớờởỡợéèẻẽẹêếềểễệúùủũụưứừửữựíìỉĩịýỳỷỹỵđ!\"\',\-\.:;?_\(\)]+$'
def latin_extract(data):
    latin_extract_data=[]
    # duyet qua tung van ban
    for i in data:
      if i == 1:
        break
      # thay the xuong dong la dau cham ket thuc
      i=i.replace("\n",".")
      # tach van ban theo dau cham ket thuc
      sentences=i.split(".")
      # re.match(alphabet, j.lower()): kiểm tra xem phiên bản viết thường j có trong alphabet hay không.
      for j in sentences:
        if len(j.split()) > 2 and re.match(alphabet, j.lower()):
            latin_extract_data.append(j)

    return latin_extract_data
```

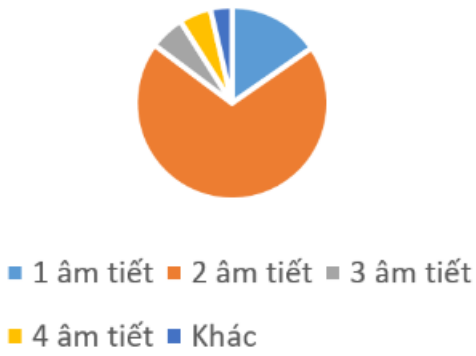
Hình 4.2: Hình ảnh dòng lệnh thực hiện việc tách câu, loại bỏ kí tự.

Sau đó, tách các cụm từ trong câu thành bi-gram (Hình 4.3, với đầu vào *phrases* là tập các câu cần tách), dữ liệu lúc này được đi qua một lớp nhiễu nhằm tạo ra lỗi chính tả, mô phỏng trường hợp sai lỗi chính tả như ngoài thực tế.

```python
def gen_ngrams(words, n=2):
    return ngrams(words.split(), n)

def generate_bi_grams(phrases):
    list_ngrams = []
    for p in tqdm(phrases):

      # neu khong nam trong alphabet thi bo qua
      if not re.match(alphabet, p.lower()):
        continue

      # tach p thanh cac bi gram
      for ngr in gen_ngrams(p, NGRAM):
        if len(" ".join(ngr)) < MAXLEN:
          list_ngrams.append(" ".join(ngr))

    return list_ngrams
```

Hình 4.3: Hình ảnh dòng lệnh thực hiện việc tách bi-gram.

Mô hình Bi-gram (N-gram có n = 2) được lựa chọn sử dụng trong mô hình này. Theo “Chữ Quốc ngữ hiện nay qua các con số thống kê”, hội thảo "Chữ Quốc ngữ", tháng 10/2015, Phú Yên của Đinh Điền, Đỗ Đức Hào, có đến 69,8% từ trong từ điển tiếng Việt có 2 âm tiết. (Hình 4.4)



Hình 4.4: Thống kê từ theo chiều dài. Nguồn: Thống kê trên từ điển tiếng Việt

Vì thế, bài tiểu luận chọn việc sử dụng bi-gram nhằm mục đích nắm bắt được các cụm từ trong từ điển, cộng với việc nếu một từ trong cụm từ bị gõ sai thì có thể sử dụng thông tin từ từ còn lại để sửa sai.

Trường hợp sai lỗi chính tả bao gồm có:

- Lỗi sai chính tả thông thường khi gõ bàn phím.
- Từ viết tắt.
- Từ địa phương.
- Teencode: Từ ngữ mới được sử dụng bởi thanh thiếu niên hiện nay.

Lớp nhiễu sẽ duyệt qua các từ trong Bi-gram. Với mỗi kí tự trong Bi-gram, lớp nhiễu sẽ sinh ra một hàm ngẫu nhiêu dựa trên tham số nhiễu đã cho từ trước. Nếu hàm ngẫu nhiên vượt qua một ngưỡng nhất định, lớp sẽ chuyển kí tự đang xét thành một kí tự tương đang trong tập kí tự nhiễu gồm teencode, lỗi sai chính tả thông thường, từ địa phương và từ viết tắt (Hình 4.5).

```
# Loi go nham
typo={"ă":"aw","â":"aa","á":"as","à":"af","ả":"ar","ã":"ax","ạ":"aj","ắ":"aws","ố":"oor","ỗ":"oox","ộ":"ooj","ơ":"ow",
"ằ":"awf","ẳ":"awr","ẵ":"awx","ặ":"awj","ó":"os","ò":"of","ỏ":"or","õ":"ox","ọ":"oj","ô":"oo","ố":"oos","ồ":"oof",
"ớ":"ows","ờ":"owf","ở":"owr","ỡ":"owx","ợ":"owj","é":"es","è":"ef","ẻ":"er","ẽ":"ex","ẹ":"ej","ê":"ee","ế":"ees","ề":"eef",
"ể":"eer","ễ":"eex","ệ":"eej","ú":"us","ù":"uf","ủ":"ur","ũ":"ux","ụ":"uj","ư":"uw","ứ":"uws","ừ":"uwf","ử":"uwr","ữ":"uwx",
"ự":"uwj","í":"is","ì":"if","ỉ":"ir","ị":"ij","ĩ":"ix","ý":"ys","ỳ":"yf","ỷ":"yr","ỵ":"yj","đ":"dd",
"Ă":"Aw","Â":"Aa","Á":"As","À":"Af","Ả":"Ar","Ã":"Ax","Ạ":"Aj","Ắ":"Aws","Ố":"Oor","Ỗ":"Oox","Ộ":"Ooj","Ơ":"Ow",
"Ằ":"AWF","Ẳ":"Awr","Ẵ":"Awx","Ặ":"Awj","Ó":"Os","Ò":"Of","Ỏ":"Or","Õ":"Ox","Ọ":"Oj","Ô":"Oo","Ố":"Oos","Ồ":"Oof",
"Ớ":"Ows","Ờ":"Owf","Ở":"Owr","Ỡ":"Owx","Ợ":"Owj","É":"Es","È":"Ef","Ẻ":"Er","Ẽ":"Ex","Ẹ":"Ej","Ê":"Ee","Ế":"Ees","Ề":"Eef",
"Ể":"Eer","Ễ":"Eex","Ệ":"Eej","Ú":"Us","Ù":"Uf","Ủ":"Ur","Ũ":"Ux","Ụ":"Uj","Ư":"Uw","Ứ":"Uws","Ừ":"Uwf","Ử":"Uwr","Ữ":"Uwx",
"Ự":"Uwj","Í":"Is","Ì":"If","Ỉ":"Ir","Ị":"Ij","Ĩ":"Ix","Ý":"Ys","Ỳ":"Yf","Ỷ":"Yr","Ỵ":"Yj","Đ":"Dd"}

# dia phuong
region={"è":"ẽ","ẽ":"è","ù":"ủ","ủ":"ù","ã":"à","à":"ã","ò":"õ","õ":"ò","i":"j"}
region2={"s":"x","l":"n","n":"l","x":"s","d":"gi","S":"X","L":"N","N":"L","X":"S","Gi":"D","D":"Gi"}

# nguyen am
vowel=list("aeiouyáàảạãâấầẩậẫăắằẳặẵóòỏọõôốồổộỗơớờởợỡéèẻẹẽêếềểệễúùủụũưứừửựữíìỉịĩýỳỷỵỹ")

# viet tat
acronym={"không":"ko"," anh":" a","em":"e","biết":"bít","giờ":"h","gì":"j","muốn":"mún","học":"hok","yêu":"iu",
         "chồng":"ck","vợ":"vk"," ông":" ô","được":"đc","tôi":"t",
         "Không":"Ko"," Anh":" A","Em":"E","Biết":"Bít","Giờ":"H","Gì":"J","Muốn":"Mún","Học":"Hok","Yêu":"Iu",
         "Chồng":"Ck","Vợ":"Vk"," Ông":" Ô","Được":"Đc","Tôi":"T",}

# teencode
teen={"ch":"ck","ph":"f","th":"tk","nh":"nk",
      "Ch":"Ck","Ph":"F","Th":"Tk","Nh":"Nk"}
```

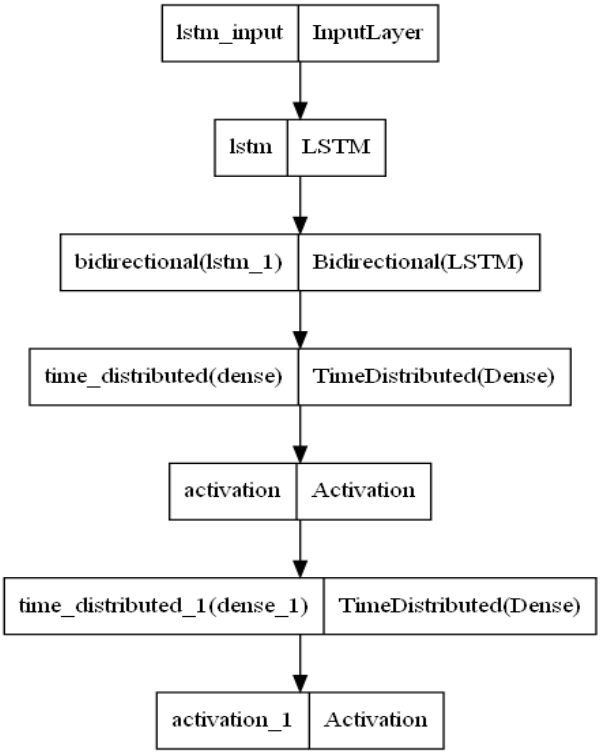
Hình 4.5: Hình ảnh các lỗi chính tả trong lớp gây nhiễu.

Ví dụ cho việc sử dụng lớp nhiễu như sau, với câu đầu tiên là câu gốc, và sâu thứ hai là câu sau khi đã đưa vào lớp nhiễu.

- Original: "Làm sạch văn bản, tách các cụm từ trong câu, nếu có kí tự đặc biệt thì xóa chúng đi"
- Sau khi Add Noise: "Làm sạch văn ảbn, tách các cụm từ trong caau, neesu có kí tự đặc biệt thì xóa chúng đi"

4.3 Xây dựng mô hình

Mô hình Bi-directional LSTM (hình 4.6) có khả năng nắm bắt được ngữ cảnh của đoạn văn bản đang xét, và lúc huấn luyện có thể lưu lại những thông tin cần thiết cho việc sửa lỗi văn bản. Đầu vào của mô hình là các chuỗi từ Bi-gram có lỗi sai chính tả và đầu ra là những Bi-gram từ đã được sửa lỗi. Đây là những lí do cho việc lựa chọn mô hình Bi-directional LSTM dạng many-to-many kết hợp với mô hình Bi-gram.



Hình 4.6: Hình ảnh mô hình mạng Bi-directional LSTM.

TimeDistributed(Dense(256)):

- Dense(256): Là một tầng mạng nơ-ron đầy đủ kết nối (fully connected layer) với 256 đơn vị (neurons) hoặc nút. Tầng này kết nối mỗi đầu vào từ tầng trước nó với mỗi đầu ra của nó.
- TimeDistributed(): Được sử dụng khi muốn áp dụng tầng mạng nơ-ron cho mỗi "thời điểm"của dữ liệu đầu vào.

Activation('ReLU') là một hàm kích hoạt được sử dụng trong các mô hình mạng nơ-ron, và ở đây, 'relu' là viết tắt của Rectified Linear Unit. Hàm kích hoạt này thường được áp dụng sau một tầng mạng nơ-ron để giúp mô hình học được các đặc trưng phi tuyến tính và tăng cường khả năng học của nó.

Activation('softmax'): Hàm softmax chuyển đầu ra của một tầng mạng nơ-ron thành một phân phối xác suất, tức là tạo ra một tập hợp các giá trị có tổng bằng 1, giúp mô hình có thể dự đoán lớp của mỗi mẫu dữ liệu.

Mô hình sẽ có tham số epoch (số lần tập huấn luyện được đưa vào mô hình) là 5. Mỗi epoch sẽ tốn 1 giờ 30 phút để huấn luyện. Tổng thời gian huấn luyện mô hình là 7 tiếng 30 phút.

Thiết lập tham số như bảng 4.7:

Tham số	Giá trị
Embedding dimension	199
Hidden dimension	256
Number of hidden layers	2
Batch size	512
Learning rate	0.001
Drop out rate	0.2

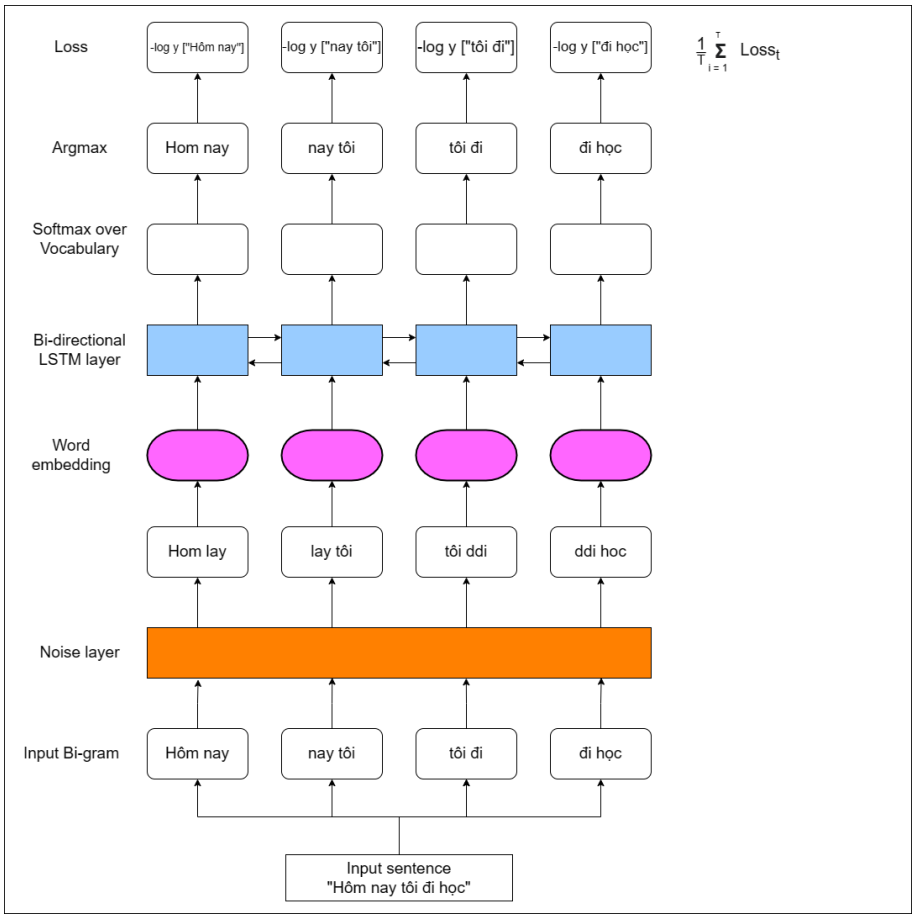
Bảng 4.7: Bảng thiết lập tham số mô hình.

Bảng 4.7 trên được lấy từ quá trình xây dựng mô hình ở hình 4.7.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 40, 256)           466944

 bidirectional (Bidirection  (None, 40, 512)           1050624
 al)

 time_distributed (TimeDist  (None, 40, 256)           131328
 ributed)

 activation (Activation)     (None, 40, 256)           0

 time_distributed_1 (TimeDi  (None, 40, 199)           51143
 stributed)

 activation_1 (Activation)   (None, 40, 199)           0

=================================================================
```

Hình 4.7: Hình ảnh thiết lập tham số mô hình.

Như vậy, chúng ta tổng kết lại được quá trình huấn luyện như hình 4.8.

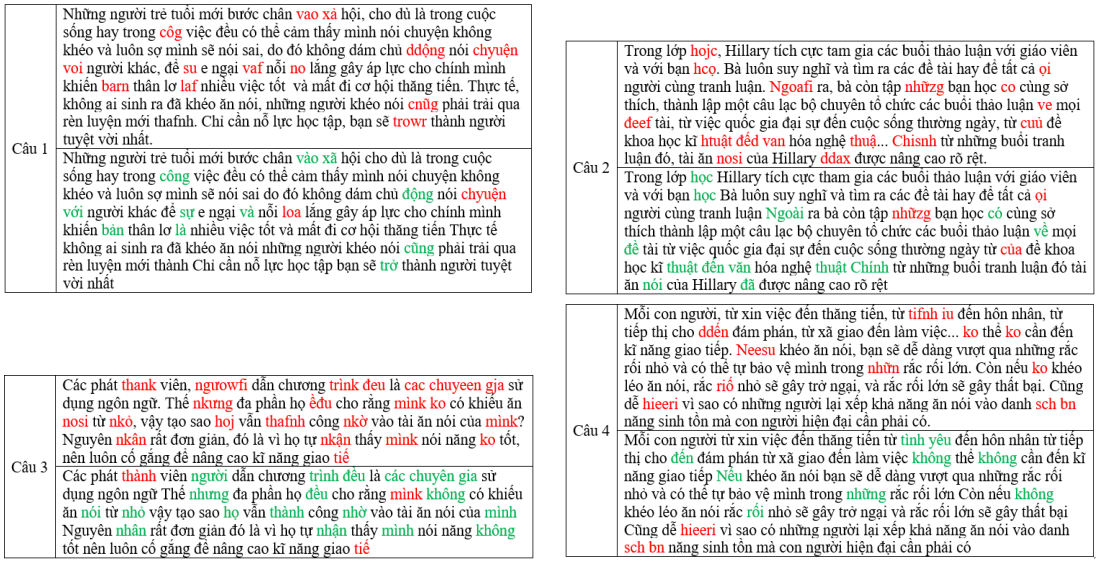


Hình 4.8: Quy trình sửa lỗi chính tả tiếng Việt sử dụng Bi-directional LSTM và Bi-gram.

4.4 Kết quả thực nghiệm và đánh giá

4.4.1 Kết quả thực nghiệm

Hình 4.9 dưới đây là một số câu văn đã được làm nhiễu, sau đó thực hiện mô hình sửa lỗi chính tả tiếng Việt.



Hình 4.9: Hình ảnh một số kết quả thực nghiệm (Ở mỗi câu, phía trên là câu văn đã được gây nhiễu, phía dưới là kết quả sau khi chạy qua mô hình sửa lỗi chính tả tiếng Việt).

Để có được cái nhìn tổng quan hơn, nhóm tiến hành so sánh mô hình đã xây dựng với mô hình của Nguyễn Thị Xuân Hương và Nguyễn Hà Thanh như bảng 4.8 dưới đây (Tham khảo [8]):

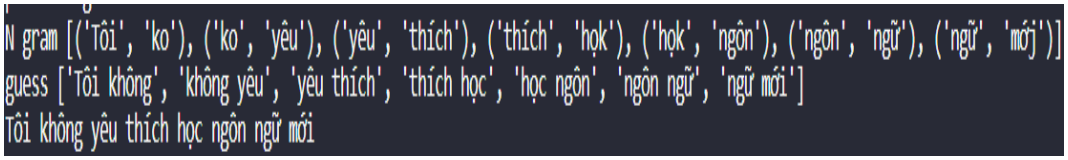
Bảng 4.8: Bảng so sánh kết quả đầu ra một số câu văn giữa mô hình của Nguyễn Thị Xuân Hương, mô hình của Nguyễn Hà Thanh và mô hình của chúng tôi.

Input sentence	**Correct sentence**	**N.T.X.Huong model output**	**N.H.Thanh model output**	**Our model output**
Lúc đó, bả ngán tôi **nắm** vì tôi ốm nhom, **nại** nhìn **như** thằng bụi đời.	**Lúc** đó, bả ngán tôi **lắm** vì tôi ốm nhom, **lại** nhìn **như** thằng bụi đời.	**Núc** đó, bả **quán** tôi lắm vì tôi ốm nhom, **nại** nhìn **dư** thằng bụi đời.	**Lúc** đó, bả ngán tôi **lắm** vì tôi ốm nhom, **lại** nhìn **như** thằng bụi đời.	**Lúc** đó bả ngán tôi **nắm** vì tôi ốm nhom **lại** nhìn **như** thằng bụi đời
Chường của **tráu** bạn **lào** cũng được **tra** mẹ đưa **se** hơi đi học.	**Trường** của **cháu** bạn **nào** cũng được **cha** mẹ đưa **xe** hơi đi học.	**Chường** của **tráu** bạn **lào** cũng được **cha** mẹ đưa **xe** hơi đi học.	**Trường** của **cháu** bạn **nào** cũng được **cha** mẹ đưa **xe** hơi đi học.	**Chường** của **trái** bạn **nào** cũng được **cha** mẹ đưa **se** hơi đi học.
Cúm A/H5N1 ở người phụ thuộc vào **rịch** cúm ở **da** cầm.	Cúm A/H5N1 ở người phụ thuộc vào **dịch** cúm ở **gia** cầm.	Cúm A/H5N1 ở người phụ thuộc vào **dịch** cúm ở **nha** cầm.	Cúm A/H5N1 ở người phụ thuộc vào **dịch** cúm ở **gia** cầm.	Cúm A H5N1 ở người phụ thuộc vào **rịch** cúm ở **da** cầm
Phó Thủ tướng, Bộ **trưởng** Ngoại giao Phạm **Da** Khiêm trả lời phỏng vấn.	Phó Thủ tướng, Bộ **trưởng** Ngoại giao Phạm **Gia** Khiêm trả lời phỏng vấn.	Phó Thủ tướng, Bộ **trưởng** Ngoại giao Phạm **Gia** Khiêm trả lời phỏng vấn.	Phó Thủ tướng, Bộ **trưởng** Ngoại giao Phạm **Ra** Khiêm trả lời phỏng vấn.	Phó Thủ tướng Bộ **trưởng** Ngoại giao Phạm **Da** Khiêm trả lời phỏng vấn

4.4.2 Đánh giá mô hình

Áp dụng lớp nhiễu cho tệp kiểm thử, đưa văn bản từ tệp kiểm thử và thêm lỗi chính tả vào chúng trước khi đưa vào mô hình. Mô hình sẽ sửa lỗi văn bản và sau đó so sánh với câu nguyên bản chưa đi qua lớp nhiễu.

Ví dụ, ta thử nghiệm đưa vào một câu là **"Tôi không yêu thích học ngôn ngữ mới**", đầu vào này sau khi qua lớp làm nhiễu tự động sẽ trở thành **"Tôi ko yêu thích họk ngôn ngữ mój"**, và kết quả khi đưa câu có lỗi chính tả qua mô hình sẽ có dạng như hình 4.10:



Hình 4.10: Kết quả khi đưa câu có lỗi chính tả vào mô hình.

Lúc này hàm đánh giá được tính theo công thức dưới đây:

$$Accuracy_{total} = \frac{n_{correctword}}{n_{totalword}} \times 100 \tag{4.1}$$

Vì không có đủ thời gian để kiểm thử toàn bộ tập dữ liệu kiểm thử, nhóm chúng tôi quyết định kiểm thử mô hình ở 5 mức (100 câu, 200 câu, ..., 500 câu) và có kết quả như bảng 4.9.

Bảng 4.9: Kết quả chính xác với các ngưỡng (số câu) dữ liệu đầu vào (lấy từ tập kiểm thử).

Test Size (câu)	100	200	300	400	500
Accuracy Value(%)	92	93.6	94.1	93.9	94.5

4.5 Thảo luận và hướng phát triển

Như vậy, bài tiểu luận đã đưa ra các lý thuyết và vấn đề trong quá trình thiết lập, huấn luyện và xây dựng một mô hình sửa lỗi chính tả cho tiếng Việt. Kết quả ban đầu đạt được là tiền đề để tạo ra các trợ lý ảo, xây dựng các ứng dụng thông minh có thể hiểu được ngôn ngữ tiếng Việt. Có khả năng áp dụng vào các bài toán thực tế, ví dụ như các gợi ý sửa lỗi chính tả trong tình soạn thảo, tự động sửa lỗi chính tả trong tìm kiếm, gợi ý từ tiếp theo trong soạn tin nhắn, . . .

Từ kết quả thực nghiệm của tiểu luận này, nhóm 6 có một số nhận xét:

- Từ kết quả thực nghiệm ở hình 4.9, bảng 4.8, bảng 4.9, có thể thấy rằng nhược điểm của việc thiếu dữ liệu là lớn như thế nào, với độ chính xác ở các mức đầu vào hầu như là lớn hơn 90 phần trăm, tuy nhiên, mô hình đã không áp dụng tốt trong một số lĩnh vực trong thực tế (những lĩnh vực không có trong tập huấn luyện như y tế, ..,). Điều này có thể dẫn đến vấn đề overfitting, cần mở rộng thêm tập dữ liệu.
- Với các chuỗi câu dài thì mạng huấn luyện mất nhiều thời gian hơn.
- Với độ đo chính xác (Accuracy) đã giải quyết mặt hạn chế do dữ liệu thu thập đầu vào không được phong phú.
- Để áp dụng vào thực tiễn, cần lấy dữ liệu thực tế thay vì tự tạo. Do tự tạo nhiễu sẽ không thể bao quát được lỗi của tiếng Việt.

Hướng phát triển của bài toán này có thể là tiếp tục kế thừa những nghiên cứu trước đây và phát triển mô hình sửa lỗi chính tả mới có khả năng sửa lỗi ở các loại khác ngoài lỗi phát âm, lỗi viết tắt, lỗi tiếng lóng, lỗi "quên bật Vietkey" còn có thể sửa lỗi theo ngữ cảnh. Áp dụng các phương pháp học sâu khác để cải thiện độ chính xác như: Attention, Beam search.

Chương 5

KẾT LUẬN

Trong bài tiểu luận này, chúng tôi đã xây dựng hệ thống sửa lỗi chính tả dựa trên mô hình phân loại đầu vào theo hướng mạng nơ ron bộ nhớ dài ngắn song song (Bidirectional Long Short-Term Memory - BiLSTM).

Bộ dữ liệu đầu vào là các văn bản được thêm nhiễu và văn bản gốc. Quá trình huấn luyện được tiến hành dựa trên kỹ thuật mạng nơ ron sâu thông qua hàm softmax để thể hiện xác suất của lớp và Entropy chéo được định nghĩa để đánh giá mục tiêu của đầu ra. Ý tưởng chính của mô hình này là sử dụng mã hóa hướng viết sai chính tả để xác định và sửa các vị trí lỗi. Mô hình có độ chính xác tốt hơn so với các phương pháp kiểm tra chính tả tiếng Việt hiện nay. Kết quả nghiên cứu này có tính ứng dụng cao trong thực tiễn.

Do thời gian và khả năng có hạn nên không thể tránh khỏi sai sót. Vì vậy, mong cô **Đoàn Thị Hồng Phước** có thể nhận xét và đưa ra ý kiến để nhóm hoàn thiện hơn bài tiểu luận của mình. Cuối cùng, nhóm xin chân thành cảm ơn cô **Đoàn Thị Hồng Phước** đã tích cực hướng dẫn trong suốt thời gian nhóm thực hiện tiểu luận này.

Tài liệu tham khảo

[1] O. Buyuk. Context-dependent sequence-to-sequence turkish spelling correction. 2017.

[2] Gu S.; Lang F. A chinese text corrector based on seq2seq model. 2020.

[3] Nguyen Duc Hai and Nguyen Pham Hanh Nhi. Syntacticparser in vietnamese sentences and its application in spellchecking. 1999.

[4] Nguyen Hua Phung; Thuan D. Ngo; Dung A. Phan; Thu P.T Dinh; Thang Q. Huynh. Vietnamese spelling detection and correction using bi-gram, minimum edit distance, soundex algorithms with some additional heuristics. 2008.

[5] Pham Dinh Khanh. Ly thuyet ve mang lstm. 2019.

[6] Zhou Y.; Porwal U.; Konow. Spelling correction as a foreign language. 2017.

[7] Nguyen Thi Xuan Huong; Dang; The Tung Nguyen; Anh Cuong Le;. Using large n-gram for vietnamese spell checking. 2015.

[8] Nguyen Ha Thanh; Dang Tran Binh; Nguyen Le Minh. Deep learning approach for vietnamese consonant misspell correction. 2019.

[9] Hoang Phe. "spelling dictionary" (in vietnamese). 2006.

[10] Hoang Phe. "vietnamese dictionary" (in vietnamese). 2006.

[11] Nguyen Hong Vu; Nguyen H.T.; Snasel. Normalization of vietnamese tweets on twitter. 2015.

[12] Dinh Truong Do; Ha Thanh Nguyen; Thang Ngoc Bui; Hieu Dinh Vo. Transformer-based model for vietnamese spelling correction. 2021.

[13] Nguyen Q.D.; Le D.A.; Zelinka. Ocr error correction for unconstrained vietnamese handwritten text. 2019.